"LOAD REBALANCING IN CLOUDS"

Smita Salunkhe [1],S.S.Sannakki[2]
1.Mtech Student, Department of Computer Science and Engineering
2.Professor, Department of Computer Science and Engineering
KLS Gogte Institute of Technology, Belgaum, Karnataka, India
Affiliated to Visvesvaraya technological University
Email:smitajs21@gmail.com,sannakkisanjeev@git.edu

Abstract: In cloud computing based applications Distributed file systems are one of the most significant and important concepts. In the distributed file systems, the computation and storage functions of the nodes are performed simultaneously; the files are split into a "number of chunks" which are allocated in various "distinct nodes" so that the tasks can be served "parallelly" over the different nodes [1].When computing takes place in clouds, failures take place frequently, and because of this the addition, replacement, upgrading of the nodes of the system takes place. The dynamic creation, deletion, appending of files of the system can occur. If this happens the imbalance of the load can take place in the following distributed file system [2]; that is, the files that are partitioned into chunks are not uniformly distributed among the nodes. In many of the production systems which have distributed file systems mainly are reliable on a centralized approach for reallocation of chunks in the nodes. If the system is completely dependent on a central node and if the distribution of files is large then there is lot of workload on this central node which surely can cause failures and bottlenecks. Here the workload is linearly proportional with the system size, because of this there can be a bottleneck in the performance and it can become a failure at a single point. In this project, a" load rebalancing algorithm" which is fully distributed is offered to deal with the imbalance of load problem. The algorithm presents a competing distributed solution compared to the techniques used in the literature. The analysis of the result shows that the system which is proposed significantly works better comparing to the distributed algorithm which were previously used in terms of factor of "load imbalance" [1], "cost of movement" [1], and "overhead" of algorithm [1].

Keywords：Load balance, distributed file systems, clouds .

I. INTRODUCTION

CLOUD Computing is a persuasive technology. Clouds can allot "their resources" when it is demanded in a dynamic way. Some of the important concepts that clouds are using are the MapReduce programming[2],distributedsystems(e.g.[3],[4],[7],[8],[9]),Virtualization". In distributed file systems[3],[4] the" load of a node" is directly proportional to file chunk number that each nodes will have[4].Because of the file creation, deletion, and appended, in clouds can be on a random basis the upgradation, replacement and addition of nodes in the file system [6] can occur, the chunks of files are not distributed as consistently as possible among the nodes. Load Balancing is an important concept in cloud computing. Here the resources should be well utilized and the performance should be maximized. The problem of load rebalancing problem[1] of the distributed file systems[3][4] takes place in huge, dynamic clouds[5].

II. LITERATURE SURVEY

"S. Ghemawat, H. Gobioff, and S.-T. Leung [3] designed and implemented the" Google File System" [3]",which is designed as a distributed file system that is very scalable for distributed Data-important applications that are large. It works on commodity hardware that is not expensive which will provides fault tolerance, and it delivers high comprehensive performance to a many number of clients. It will have many of the similar goals as previous distributed file systems [3], the design of the GFS has been made by both present and future observations of application workloads and technologies, thus reflecting a prominent difference from previous file systems.

"A. Rowstron and P. Druschel [9] presents the design and evaluation of "Pastry"[9]", it represents a routing, scalable, object location that is distributed and routing which is used for peer-to-peer applications in wide area networks. Pastry uses the internet where there is application-level routing and locates objects in a very huge overlay network of nodes . Pastry supports a large number of applications that are peer-to-peer, which includes a data

storage that is global, sharing of data , communication in groups and naming.

P. Ganesan, M. Bawa, and H. Garcia-Molina[13] **,** presented the "Balancing of Range-Partitioning Applications Data for Peer-to-Peer Systems Online,"[13] "this system solves the problem by partitioning horizontally a relation which is dynamic among a huge number of disks/nodes by using the concept of partitioning in range[13]. This type of partitioning is most often used in very big-scale parallel databases and it is also desirable in "peer-to-peer (P2P) systems[11]".

III. EXISTING SYSTEM

File	Chunks
F1	5
F2	4
F3	7
Total	16

Files and its Chunks

N1	N2	N3	N4
F3-C4			
F3-C0	F3-C5	F3-C6	
F2-C0	F3-C1	F3-C2	F3-C3
F1-C4	F2-C1	F2-C2	F2-C3
F1-C0	F1-C1	F1-C2	F1-C3
4	3	3	2

Chunk distribution of files in nodes in the existing Sytem.

Table 1 shows the files and the distribution of its respective number of chunks .The file f1is partitioned into 5 chunks, file f2 is partitioned into 4 chunks and f3 into 7 chunks. Therefore the files are divided into 16 chunks.

Table 2 shows the imbalance in the existing system when the distribution of these chunks takes place. It shows that there are 4 nodes and among them distribution of these chunks take place not in a uniform manner thus causing imbalance in the load .This affects the resource utilization, movement cost, load imbalance factor and convergence to an extended rate.

IV. PROPOSED SYSTEM

Table 1 shows the files and the distribution of its respective number of chunks .The file f1is partitioned into 5 chunks, file f2 is partitioned into 4 chunks and f3 into 7 chunks. Therefore the files are divided into 16 chunks.

File	Chunks
F1	5
F2	4
F3	7
Total	16

Files and its Chunks

not balance dimensionally. If you must use mixed units, clearly state the units for each quantity in an equation.

N1	N2	N3	N4
F2-C3	F3-C4	F3-C5	F3-C6
F2-C3	F3-C0	F3-C1	F3-C2
F1-C4	F2-C0	F2-C1	F2-C2
F1-C0	F1-C1	F1-C2	F1-C3
4	4	4	4

chunk distribution of files in nodes in the proposed System

Table 2 shows the balance in the proposed system when the distribution of these chunks takes place. It shows that there are 4 nodes and among them distribution of these chunks takes place in a uniform manner thus causing "load rebalance". According to table each node has equal number of chunks thus avoiding excess load on only some nodes.

Advantages of Proposed System are as follows:

- Deploy wide-range and failure error domain
- Reduce Network Traffic or Movement Cost
- Maximize the Network Bandwidth
- Improve overall System Performance
- Utilizes Physical Network Locality
- Better throughput and response time
- System consistency (i.e., avoid data loss)

- Excellent security
- Picking lead of node heterogeneity
- Extends resource utilization
- Speedy & Diminishes time consistency

V. IMPLEMENTATION

Elements which are declared in the interface. There are 4 modules in this project:

1. Namenode (Keep Track of Name of DataNodes)
2. Datanodes (Storage servers)
3. Client
4. Load Balancer.

NAMENODE

Namenodes are defined as the nodes which manage the metadata information. The file system namespace is manually and statically partitioned to a number of namenodes. But , as the load of work experienced by the namenodes can alter over time and if no adaptive workload technique or any migration scheme is presented to balance the loads among the namenodes, then any of the namenodes can become the performance bottleneck[6].

DATANODES (STORAGE SERVERS)

A **server** can be a system which can be used in software or in a computer hardware that can be suitable which will respond to the requests in network of computers or provides a network service. Servers are used to run on a dedicated computer, or it can be used as hosted servers in a networked environment. In many examples, a computer can provide several services and can have many of the servers running simultaneously. Various systems use the concept of client / server networking model which can include Web sites and email services[10].

In computing environment, a file server is a processor connected to a network that has the main purpose of providing shared disk accesses, for e.g. having shared storage of computer files (it can be databases, sound files, photographs, documents, images.) that are accessible by the stations that are connected to one of the same computer network. In the client server model the clients use the storage. A server of a file is not used to perform jobs of computation, and does not need to run programs as a part of its clients. It is designed mainly to allow the storage or the retrieval of data during and when the computation is carried out by the Load Balancer.

CLIENT

A client will consist of a computer program in which the operation depends on sending a request to other computer program. The example can be of; web browsers that are clients that can connect to the servers of the web and can retrieve the web pages for display. Another example can be of the email clients that can retrieve emails from the mail servers[11]. The online chat system uses various types of clients, which will differ and which will depend on the chat protocol that can be used. The online video games can also work as a client on each of the computers. The phrase "client" can be used for the computers or any of the devices that can run the software of the client or the users that can use the client software.

The clients are a component of a client–server model, this technology is used till date. Clients and servers can be computer programs which are run on the same machine and the connection can take place via inter-process communication techniques, such as with sockets used in the Internet, the programs can connect to the services operating on a perhaps remote system by using some of the Internet protocol suites.

LOAD BALANCER

The Load balancer is defined as a standalone load-balancing server (also called as balancer) which works to rebalance the load of the storage nodes[1]. The servers gets the global information of the file chunks distributed in the system from the namenodes that are used to manage the data about data(“metadata”) of complete file system. Depending on the globally acquired knowledge, it will partition the node set into two subsets, in which one (denoted by O) consists of “overloaded nodes”[1], and another (denoted by U)[1] contains the “underloaded” nodes. According to the concept the balancer usually will in random select one “heavy node” from “O” and one from a node which is light i.e. is from “U” to reallocate the loads. The reallocation will terminate if the balancer does not find a pair of” heavy node” and “light nodes” for reallocation of their loads. especially, to take care of “physical network locality” hence reducing the network traffic[5] .

The process of load balancing is to reallocate the file chunks of the files as uniformly as possible taking care of the cost of movement, the overheads of the algorithms used ,the convergence rate .The process also has to take care of the load imbalance factor. The algorithm is NP hard [12] and knapsack[12].Considering 2 nodes N1 and N2 ,Consider N1 is the node with minimum load and N2 is the load with maximum load. If the difference between the 2 nodes is greater than one then the transfer of chunks have to take place by placing the chunks uniformly or equally across the nodes. This process also has to take care of physical network locality.

1. Maintain to Metadata About Chunk loaded/removed from Data nodes
2. Find Number of Chunks in each data node
3. The Node from Heavy Load data node , Move to Light Load data Nodes
4. Update the Metadata
5. If Difference if load in Data Nodes > 2 goto step 3
6. Stop

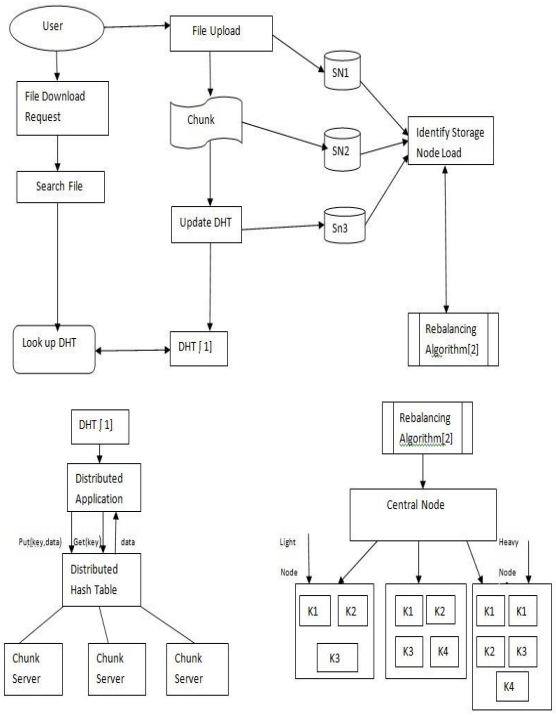

Figure: Class Diagram of complete project

VI. RESULTS

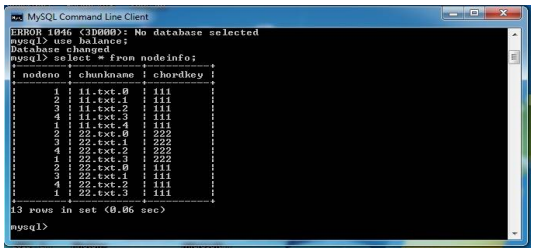

Figure: Table information by MySQL

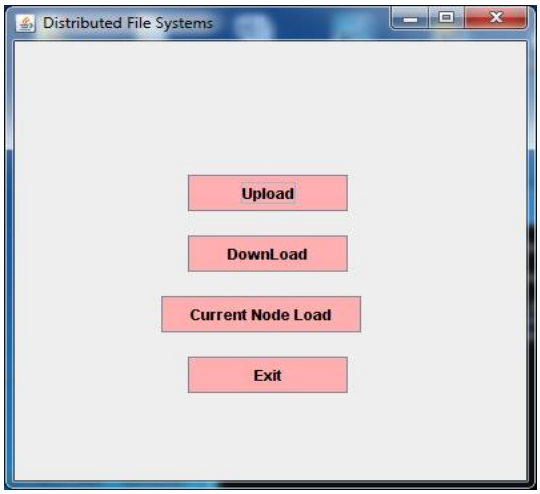

Figure: File Upload

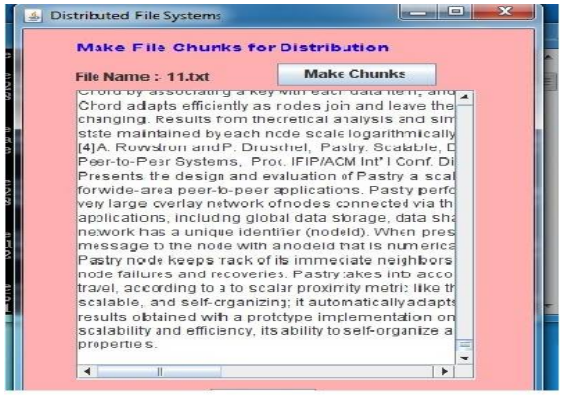

Figure: File chunk

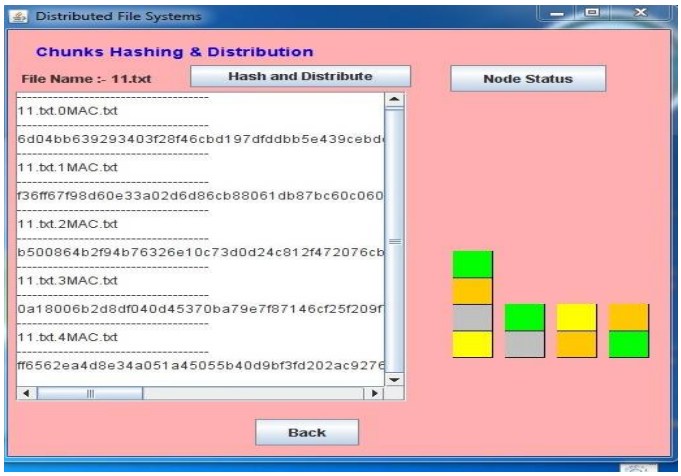

Figure : Hash and Distribute

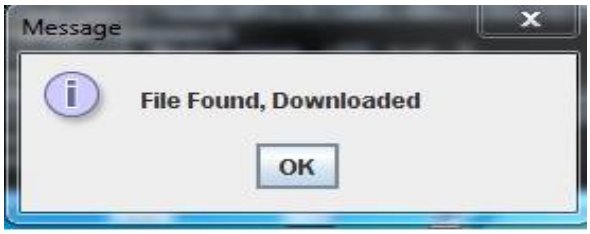

Figure : File Downloaded

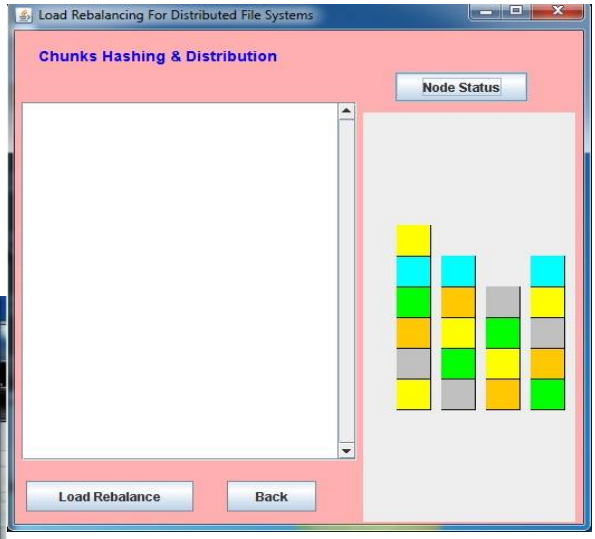

Figure :Load Imbalance

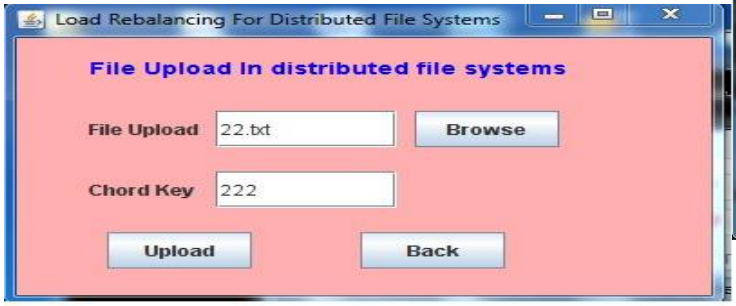

Figure : Chord Key.

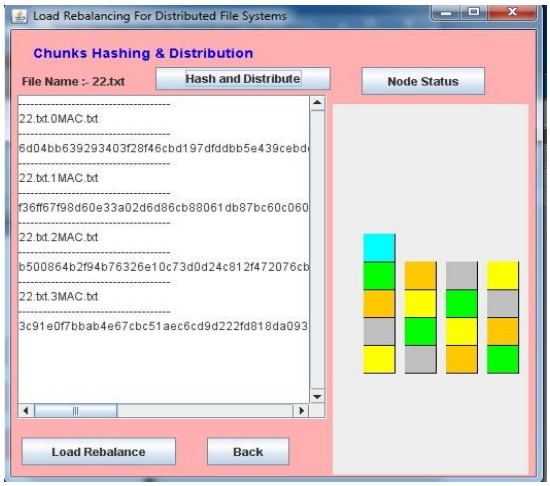

Figure : Node Status

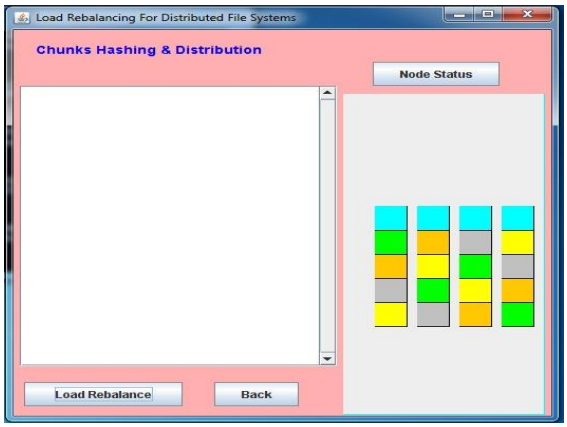

Figure :Load balance

VII. CONCLUSION

Load rebalancing for distributed file systems is specially used for big-scale, dynamic and intensive clouds of data that reallocates the file chunks in a uniform manner among the nodes such that none of

the node manage extra number of chunks and aims to reduce traffic of the network (or cost of movement), imbalance factor of load and overhead of algorithm

REFERENCES

[1]Hung-Chang Hsiao, Member, IEEE Computer Society, Hsueh-Yi Chung, Haiying Shen, Member, IEEE, and Yu-Chang Chao, "Load Rebalancing for Distributed File Systems in Clouds", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 24, NO. 5, MAY 2013.

[2] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters,"Proc. Sixth Symp. Operating System Design and Implementation (OSDI '04), pp. 137-150,Dec.2004.

[3] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," Proc. 19th ACM Symp. Operating Systems Principles (SOSP '03),pp. 29-43, Oct. 2003.

[4] Kun Liu , Gaochao Xu , and June Yuan , "An Improved Hadoop Data Load Balancing Algorithm" Journal of Networks, VOL. 8, NO. 12, DECEMBER 2013.

[5] Apache Hadoop, http://hadoop.apache.org/, 2012.

[6] Hadoop Distributed File System "Rebalancing Blocks," http://developer.yahoo.com/hadoop/tutorial/module2.html#rebalancing, 2012.

[7] K. McKusick and S. Quinlan, "GFS: Evolution on Fast-Forward," Comm. ACM,vol. 53, no. 3, pp. 42-49, Jan. 201050, Dec. 2004.

[8]I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek,F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications,"IEEE/ACM Trans. Networking, vol. 11, no. 1, pp. 17-21, Feb. 2003.

[9]A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," Proc. IFIP/ACM Int'l Conf. Distributed Systems Platforms Heidelberg, pp. 161-172, Nov. 2001.

[10]G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A.Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W.Vogels, "Dynamo: Amazon's Highly Available Key-Value Store,"Proc. 21st ACM Symp. Operating Systems Principles (SOSP '07),pp. 205-220, Oct. 2007.

[11] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica,"Load Balancing in Structured P2P Systems," Proc. Second Int'Workshop Peer-to-Peer Systems (IPTPS '02), pp. 68-79, Feb. 2003.

[12] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Co., 1979.

[13] P. Ganesan, M. Bawa, and H. Garcia-Molina, "Online Balancing of Range-partitioned Data with Applications to Peer-to-Peer Systems,"Proc. 13th Int'l Conf. Very Large Data Bases (VLDB '04),pp. 444-455, Sept. 2004.

About Authors:

Smita Salunkhe ,is an Mtech in computer Science& engg., at KLS's GI.T,Belgaum and worked as a lecturer in KLS's VPP,Belgaum., Her areas of interest are Data Mining, Data Warehousing, Cloud computing, Network security.

Prof S.S.Sannakki works at KLS's G.I.T,Belgaum, as a professor in Computer Science and engg. Department,guided many mtech students for their project work.